

A Quadratic Programming Relaxation Approach to Compute-and-Forward Network Coding Design

Baojian Zhou, *Student Member, IEEE*, Jinming Wen,
and Wai Ho Mow, *Senior Member, IEEE*

Abstract

Using physical layer network coding, compute-and-forward is a promising relaying scheme that effectively exploits the interference between users and thus achieves high rates. In this paper, we consider the problem of finding the optimal integer-valued coefficient vector for a relay in the compute-and-forward scheme to maximize the computation rate at that relay. Although this problem turns out to be a shortest vector problem, which is suspected to be NP-hard, we show that it can be relaxed to a series of equality-constrained quadratic programmings. The solutions of the relaxed problems serve as real-valued approximations of the optimal coefficient vector, and are quantized to a set of integer-valued vectors, from which a coefficient vector is selected. The key to the efficiency of our method is that the closed-form expressions of the real-valued approximations can be derived with the Lagrange multiplier method. Numerical results demonstrate that compared with the existing methods, our method offers comparable rates at an impressively low complexity.

Index Terms

physical layer network coding, AWGN networks, compute-and-forward, quadratic programming, Lagrange multiplier.

Baojian Zhou and Wai Ho Mow are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, NT, Hong Kong (e-mail: {bzhouab, eewhmow}@ust.hk). Baojian Zhou was supported by a grant from University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08).

Jinming Wen is with the Laboratoire de l'Informatique du Parallélisme, (CNRS, ENS de Lyon, Inria, UCBL), Université de Lyon, Lyon 69007, France (e-mail: jwen@math.mcgill.ca). Jinming Wen was supported in part by ANR through the HPAC project under Grant ANR 11 BS02 013.

I. INTRODUCTION

Being a promising relaying strategy in wireless networks, compute-and-forward (CF) has attracted a lot of research interest since it was proposed in 2008 by Nazer and Gastpar [1]. The advantage of CF is that it achieves higher rates in the medium signal-to-noise ratio (SNR) regime when compared with other relaying strategies, e.g., amplify-and-forward, decode-and-forward. Relays in CF attempt to decode integer linear combinations of the transmitted codewords, rather than the codewords themselves. The integer coefficient vectors corresponding to the linear combinations and the decoded messages are then forwarded to the destination. Under certain conditions (see [2] for details), the destination can recover the original source messages with enough forwarded messages and coefficient vectors from the relay.

The design of the CF scheme lies in selecting the coefficient vectors at the relays. There are many choices of coefficient vectors for one relay, and each may render a different *computation rate* [2] at that relay. Computation rate is defined as the maximum transmission rate from the associated sources to a relay such that the linear combinations at the relay can be successfully decoded. If the coefficient vectors are linearly independent, the *achievable rate* of the network equals the minimum computation rate; otherwise, the achievable rate of the network is zero and none of the messages can be recovered. The objective of designing the CF scheme is to maximize the overall achievable rate of the network. One approximation method is presented in [3]. However, for those networks where each relay is allowed to send only one coefficient vector to the destination, and only local channel state information (CSI) is available, i.e., each relay knows merely its own channel vector, one reasonable solution is to select the coefficient vector that maximizes the computation rate at each relay.

In this paper, we consider additive white Gaussian noise (AWGN) networks where only local CSI is available, and focus on the CF network coding design problem with the objective being maximizing the computation rate at a relay by choosing the optimal coefficient vector. It has been shown that this problem reduces to a shortest vector problem (SVP). Different methods have been developed to tackle this problem. The branch-and-bound method proposed in [4] finds the optimal solution but its efficiency degrades as the dimension of channel vectors grows according to the simulation results. Although the general SVP is suspected to be NP-hard, Sahraei and Gastpar showed in [5] that the SVP in the CF design is special, and developed an algorithm

(called the “SG” method in this paper) that solves the SVP in polynomial time. For independent and identically distributed (i.i.d.) Gaussian channel entries, the complexity of the SG method is of order 2.5 with respect to the dimension, and is linear with respect to the square root of the signal-to-noise ratio (SNR). A class of methods are those based on lattice reduction (LR) algorithms (e.g., Minkowski, HKZ, LLL, and CLLL LR algorithms; c.f. [6]–[11]). The method in [12] based on the LLL LR algorithm [7] provides close-to-optimal rates and is well-known to be of polynomial time complexity with respect to the vector dimension. In [13], we proposed an efficient method based on sphere decoding to find the optimal coefficient vector; however there is no theoretical guarantee on the complexity.

Our goal in this work is to develop a new method that finds a suboptimal coefficient vector for a relay with low complexity compared with the existing methods, while provides a close-to-optimal computation rate at the same time. Taking advantage of some useful properties of the problem, we first show that the original SVP can be approximated by a series of quadratic programmings (QPs). The closed-form solutions of the QPs are derived by use of the Lagrange multiplier method and can be computed with linear complexity with respect to the dimension, which is the key to the efficiency of our method. The solutions of the QPs serve as real-valued approximations of the integer coefficient vector, and are quantized into a set of candidate integer vectors by a successive quantization algorithm. Finally, the integer vector in the candidate set that maximizes the computation rate is selected to be the coefficient vector. The complexity of our method is of order 1.5 with respect to the dimension for i.i.d. Gaussian channel entries, and is lower than the above mentioned methods. Numerical results demonstrate that among existing methods that provide close-to-optimal rates, our method is much more efficient as expected.

As a summary, our contributions in this work include the following:

- For the real-valued channels, we develop a quadratic programming relaxation approach to find a suboptimal coefficient vector for a relay so that the computation rate at that relay is close-to-optimal. The complexity is $O(L\sqrt{P\|\mathbf{h}\|^2})$ for a given channel vector $\mathbf{h} \in \mathbb{R}^L$ and signal power constraint P , and is of average value $O(P^{0.5}L^{1.5})$ for i.i.d. standard Gaussian channel entries.
- For the complex-valued channels, we demonstrate how to apply our method in an efficient way to find the complex-valued coefficient vector.
- Extensive simulation results are presented to compare the effectiveness and efficiency of

our method with the existing methods.

Part of this work has been presented in [14]. One main improvement here is the complexity order for i.i.d. Gaussian channel entries is further reduced from 3 to 1.5.

In the following, we will first introduce the system model of AWGN networks as well as the CF network coding design problem in Section II. Then in Section III, we will present our proposed method in detail. Numerical results will be shown in Section IV. Finally, we will conclude our work in Section V.

Notation. Let \mathbb{R} be the real field, \mathbb{C} be the complex field, and \mathbb{Z} be the ring of integers. Boldface lowercase letters denote column vectors, and boldface uppercase letters denote matrices, e.g., $\mathbf{w} \in \mathbb{R}^L$ and $\mathbf{W} \in \mathbb{R}^{M \times L}$. $\|\mathbf{w}\|$ denotes the ℓ^2 -norm of \mathbf{w} , and \mathbf{w}^T denotes the transpose of \mathbf{w} . For a vector \mathbf{w} , let $\mathbf{w}(\ell)$ be the element with index ℓ , and $\mathbf{w}(i:j)$ be the vector composed of elements with indices from i to j . For a matrix \mathbf{W} , let $\mathbf{W}(i:j, k:\ell)$ be the submatrix containing elements with row indices from i to j and column indices from k to ℓ , $\mathbf{W}(i:j, k)$ be the submatrix containing elements with row indices from i to j and column index k , $\mathbf{W}(i, k:\ell)$ be the submatrix containing elements with row index i and column indices from k to ℓ , and $\mathbf{W}(i, j)$ be the element with row index i and column index j . Let $\lfloor x \rfloor$ and $\lceil x \rceil$, i.e., the corresponding floor and ceiling functions of x , be the maximum integer no greater than x and the minimum integer no less than x , respectively. Let $\lfloor \mathbf{w} \rfloor_\ell$ and $\lceil \mathbf{w} \rceil_\ell$ be the vectors generated from \mathbf{w} by applying the corresponding operation on the ℓ -th element only. $\mathbf{0}$ denotes an all-zero vector, and \mathbf{I} denotes an identity matrix. $\text{sign}(\mathbf{w})$ returns the vector that contains the signs of the elements in \mathbf{w} . $\text{abs}(\mathbf{w})$ returns the vector whose elements are the absolute values of the elements in \mathbf{w} .

II. PROBLEM STATEMENT

We consider additive white Gaussian noise (AWGN) networks [2] where sources, relays and destinations are connected by linear channels with AWGN. For the ease of explanation, we first develop our method for real-valued channels, and then demonstrate how to apply our method to complex-valued channels. An AWGN network with real-valued channels is defined as the following.

Definition 1. (Real-Valued Channel Model) *In an AWGN network, each relay (indexed by $m =$*

$1, 2, \dots, M$) observes a noisy linear combination of the transmitted signals through the channel,

$$\mathbf{y}_m = \sum_{\ell=1}^L \mathbf{h}_m(\ell) \mathbf{x}_\ell + \mathbf{z}_m, \quad (1)$$

where $\mathbf{x}_\ell \in \mathbb{R}^n$ with the power constraint $\frac{1}{n} \|\mathbf{x}_\ell\|^2 \leq P$ is the transmitted codeword from source ℓ ($\ell = 1, 2, \dots, L$), $\mathbf{h}_m \in \mathbb{R}^L$ is the channel vector to relay m , $\mathbf{h}_m(\ell)$ is the ℓ -th entry of \mathbf{h}_m , $\mathbf{z}_m \in \mathbb{R}^n$ is the noise vector with entries being i.i.d. Gaussian, i.e., $\mathbf{z}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and \mathbf{y}_m is the signal received at relay m .

In the sequel, we will focus on one relay and thus ignore the subscript “ m ” in \mathbf{h}_m , \mathbf{a}_m , etc.

In CF, rather than directly decode the received signal \mathbf{y} as a codeword, a relay first applies to \mathbf{y} an amplifying factor α such that $\alpha \mathbf{h}$ is close to an integer coefficient vector \mathbf{a} , and tries to decode $\alpha \mathbf{y}$ as an integer linear combination, whose coefficients form \mathbf{a} , of the original codewords $\{\mathbf{x}_\ell\}$. The *computation rate* [2] is the maximum transmission rate from the associated sources to a relay such that the integer linear combinations at the relay can be decoded with arbitrarily small error probability. Assume the log function is with respect to base 2, and define $\log^+(w) \triangleq \max(\log(w), 0)$. The computation rate can be calculated with Theorem 1 from [2].

Theorem 1. (Computation Rate in Real-Valued Channel Model) *For a relay with coefficient vector \mathbf{a} in the real-valued channel model defined in Definition 1, the following computation rate is achievable,*

$$\mathcal{R}(\mathbf{h}, \mathbf{a}) = \frac{1}{2} \log^+ \left(\left(\|\mathbf{a}\|^2 - \frac{P(\mathbf{h}^T \mathbf{a})^2}{1 + P \|\mathbf{h}\|^2} \right)^{-1} \right). \quad (2)$$

With the computation rate being the metric, we define the optimal coefficient vector as follows.

Definition 2. (The Optimal Coefficient Vector) *The optimal coefficient vector \mathbf{a}^* for a channel vector \mathbf{h} is the one that maximizes the computation rate,*

$$\mathbf{a}^* = \arg \max_{\mathbf{a} \in \mathbb{Z}^L \setminus \{\mathbf{0}\}} \mathcal{R}(\mathbf{h}, \mathbf{a}). \quad (3)$$

After a few simple manipulations, the optimization problem stated in (3) can be written in

the following quadratic form [3],

$$\mathbf{a}^* = \arg \min_{\mathbf{a} \in \mathbb{Z}^L \setminus \{\mathbf{0}\}} \mathbf{a}^T \mathbf{G} \mathbf{a}, \quad (4)$$

where

$$\mathbf{G} \triangleq \mathbf{I} - \frac{P}{1 + P \|\mathbf{h}\|^2} \mathbf{h} \mathbf{h}^T. \quad (5)$$

If we take \mathbf{G} , which is positive definite, as the *Gram matrix* of a lattice Λ , then the problem turns out to be the SVP in the lattice Λ . In the next section, we will propose an efficient approximation method based on QP relaxation that gives a suboptimal coefficient vector.

III. PROPOSED METHOD

In this section, we will first derive our method for the real-valued channel model, and then extend the method for the complex-valued channel model.

A. Preliminaries

We start with investigating some properties of the problem, which is the basis of our new method.

Definition 3. (*Signature Matrix*) A signature matrix is a diagonal matrix whose diagonal elements are ± 1 .

Definition 4. (*Signed Permutation Matrix*) A signed permutation matrix is a generalized permutation matrix whose nonzero entries are ± 1 .

Remark 1. After replacing -1 's with 1 's, a signed permutation matrix becomes a permutation matrix. Obviously, signed permutation matrices are unimodular and orthogonal. Every signed permutation matrix can be expressed as $\mathbf{S} = \mathbf{P}\mathbf{T}$, where \mathbf{T} is a signature matrix, and \mathbf{P} is a permutation matrix.

Theorem 2. If \mathbf{a}^* is the optimal coefficient vector for a channel vector \mathbf{h} with power constraint P , then for any signed permutation matrix $\mathbf{S} \in \mathbb{Z}^{L \times L}$, $\mathbf{S}\mathbf{a}^*$ is optimal for $\mathbf{S}\mathbf{h}$ with the same power constraint P , and $\mathcal{R}(\mathbf{h}, \mathbf{a}^*) = \mathcal{R}(\mathbf{S}\mathbf{h}, \mathbf{S}\mathbf{a}^*)$.

Proof: We first show $\mathcal{R}(\mathbf{h}, \mathbf{a}) = \mathcal{R}(\mathbf{Sh}, \mathbf{Sa})$ for any \mathbf{h} and \mathbf{a} with the same power constraint P . \mathbf{S} is unimodular, then \mathbf{Sa} is an integer vector and can be applied as a coefficient vector. \mathbf{S} is orthogonal, then $\mathbf{S}^T \mathbf{S} = \mathbf{I}$. $\|\mathbf{Sh}\|^2 = \mathbf{h}^T \mathbf{S}^T \mathbf{S} \mathbf{h} = \mathbf{h}^T \mathbf{h} = \|\mathbf{h}\|^2$, and similarly $\|\mathbf{Sa}\|^2 = \|\mathbf{a}\|^2$. $(\mathbf{Sh})^T \mathbf{Sa} = \mathbf{h}^T \mathbf{S}^T \mathbf{Sa} = \mathbf{h}^T \mathbf{a}$. According to Theorem 1, the computation rate $\mathcal{R}(\mathbf{h}, \mathbf{a})$ is determined by P , $\|\mathbf{h}\|^2$, $\|\mathbf{a}\|^2$, and $\mathbf{h}^T \mathbf{a}$. Thus, $\mathcal{R}(\mathbf{h}, \mathbf{a}) = \mathcal{R}(\mathbf{Sh}, \mathbf{Sa})$.

\mathbf{a}^* is optimal for \mathbf{h} means \mathbf{a}^* maximizes $\mathcal{R}(\mathbf{h}, \mathbf{a})$. Then \mathbf{Sa}^* maximizes $\mathcal{R}(\mathbf{Sh}, \mathbf{Sa})$ since $\mathcal{R}(\mathbf{h}, \mathbf{a}) = \mathcal{R}(\mathbf{Sh}, \mathbf{Sa})$ always holds. Therefore, \mathbf{Sa}^* is optimal for \mathbf{Sh} with the same power constraint P . ■

Definition 5. (*Nonnegative Ordered Vector*) A vector \mathbf{h} is said to be nonnegative ordered if its elements are nonnegative and in nondecreasing order according to their indices.

Lemma 1. For any vector \mathbf{h} , there exists a signed permutation matrix \mathbf{S} such that \mathbf{Sh} is nonnegative ordered.

Remark 2. To find such an \mathbf{S} in Lemma 1, we can simply choose $\mathbf{S} = \mathbf{PT}$, where \mathbf{T} is a signature matrix that converts all the elements in \mathbf{h} to nonnegative, and \mathbf{P} is a permutation matrix that sorts the elements in \mathbf{Th} in nondecreasing order.

With Theorem 2 and Lemma 1, for any channel vector \mathbf{h} , we can first find a signed permutation matrix \mathbf{S} and transform \mathbf{h} to the nonnegative ordered $\bar{\mathbf{h}} = \mathbf{Sh}$, then obtain the optimal coefficient vector $\bar{\mathbf{a}}^*$ for $\bar{\mathbf{h}}$, and finally recover the desired optimal coefficient vector $\mathbf{a}^* = \mathbf{S}^{-1} \bar{\mathbf{a}}^*$ for \mathbf{h} . In this way, it suffices to focus on solving the problem in (4) for nonnegative ordered channel vectors $\bar{\mathbf{h}}$.

Remark 3. In implementation, there is no need to use the signed permutation matrix \mathbf{S} . It is merely necessary to: 1) record the sign of the elements in \mathbf{h} with a vector $\mathbf{t} = \text{sign}(\mathbf{h})$, and 2) sort $\bar{\mathbf{h}} = \text{abs}(\mathbf{h})$ in ascending order as $\bar{\mathbf{h}}$ and record the original indices of the elements with a vector \mathbf{p} such that $\bar{\mathbf{h}}(\ell) = \bar{\mathbf{h}}(\mathbf{p}(\ell))$, $\ell = 1, 2, \dots, L$. After $\bar{\mathbf{a}}^*$ for $\bar{\mathbf{h}}$ is obtained, \mathbf{a}^* for \mathbf{h} can be recovered with $\mathbf{a}^*(\mathbf{p}(\ell)) = \mathbf{t}(\mathbf{p}(\ell)) \bar{\mathbf{a}}^*(\ell)$, $\ell = 1, 2, \dots, L$.

Example 1. Given a channel vector as $\mathbf{h} = [-1.9, 0.1, 1.1]^T$, then $\mathbf{t} = [-1, 1, 1]^T$, $\text{abs}(\mathbf{h}) = [1.9, 0.1, 1.1]^T$, $\bar{\mathbf{h}} = [0.1, 1.1, 1.9]^T$, and $\mathbf{p} = [2, 3, 1]^T$. If for certain power P , $\bar{\mathbf{a}}^* = [0, 1, 2]^T$, then $\mathbf{a}^* = [-2, 0, 1]^T$.

According to (4), if \mathbf{a}^* is optimal for \mathbf{h} , then $-\mathbf{a}^*$ is also optimal for \mathbf{h} . To reduce redundancy, we restrict the optimal coefficient vector \mathbf{a}^* to be the one such that $\mathbf{h}^T \mathbf{a}^* \geq 0$ in the following.

Lemma 2. *If all the elements in a channel vector \mathbf{h} are nonnegative, then all the elements in the optimal coefficient vector \mathbf{a}^* are also nonnegative.*

Proof: Suppose $\mathbf{a}^*(i) < 0$, and define \mathbf{a}' as: $\mathbf{a}'(i) = 0$, and $\mathbf{a}'(\ell) = \mathbf{a}^*(\ell)$, $\forall \ell \neq i$. Obviously, $\|\mathbf{a}'\| < \|\mathbf{a}^*\|$, and $\mathbf{h}^T \mathbf{a}' \geq \mathbf{h}^T \mathbf{a}^* \geq 0$. Then according to (2), $\mathcal{R}(\mathbf{h}, \mathbf{a}') > \mathcal{R}(\mathbf{h}, \mathbf{a}^*)$, which implies \mathbf{a}^* is not optimal and leads to a contradiction. Thus, all the elements in \mathbf{a}^* must be nonnegative. ■

Lemma 3. *For a channel vector \mathbf{h} and its corresponding optimal coefficient vector \mathbf{a}^* , if $\mathbf{h}(i) = 0$, then $\mathbf{a}^*(i) = 0$.*

Proof: Suppose $\mathbf{h}(i) = 0$, and $\mathbf{a}^*(i) \neq 0$. Define \mathbf{a}' as: $\mathbf{a}'(i) = 0$, and $\mathbf{a}'(\ell) = \mathbf{a}^*(\ell)$, $\forall \ell \neq i$. Obviously, $\|\mathbf{a}'\| < \|\mathbf{a}^*\|$, and $\mathbf{h}^T \mathbf{a}' = \mathbf{h}^T \mathbf{a}^* \geq 0$. Then according to (2), $\mathcal{R}(\mathbf{h}, \mathbf{a}') > \mathcal{R}(\mathbf{h}, \mathbf{a}^*)$, which implies \mathbf{a}^* is not optimal. Thus, if $\mathbf{h}(i) = 0$, then $\mathbf{a}^*(i) = 0$. ■

Lemma 4. *For a channel vector \mathbf{h} and its corresponding optimal coefficient vector \mathbf{a}^* , if $\mathbf{h}(i) = \mathbf{h}(j)$, $i < j$, then $\mathbf{a}^*(i) = \mathbf{a}^*(j)$ or $\text{abs}(\mathbf{a}^*(i) - \mathbf{a}^*(j)) = 1$.*

Proof: Without loss of generality, assume $\mathbf{a}^*(i) - \mathbf{a}^*(j) < -1$. Define \mathbf{a}' as: $\mathbf{a}'(i) = \mathbf{a}^*(i) + 1$, $\mathbf{a}'(j) = \mathbf{a}^*(j) - 1$, and $\mathbf{a}'(\ell) = \mathbf{a}^*(\ell)$, $\forall \ell \notin \{i, j\}$. Obviously, $\|\mathbf{a}'\| < \|\mathbf{a}^*\|$, and $\mathbf{h}^T \mathbf{a}' = \mathbf{h}^T \mathbf{a}^* \geq 0$. Then according to (2), $\mathcal{R}(\mathbf{h}, \mathbf{a}') > \mathcal{R}(\mathbf{h}, \mathbf{a}^*)$, which implies \mathbf{a}^* is not optimal. Thus, $\mathbf{a}^*(i) - \mathbf{a}^*(j) \geq -1$. Similarly, $\mathbf{a}^*(j) - \mathbf{a}^*(i) \geq -1$. Therefore, $\mathbf{a}^*(i) = \mathbf{a}^*(j)$ or $\text{abs}(\mathbf{a}^*(i) - \mathbf{a}^*(j)) = 1$. ■

Remark 4. *In Lemma 4, for the case where $\mathbf{h}(i) = \mathbf{h}(j)$ with $\text{abs}(\mathbf{a}^*(i) - \mathbf{a}^*(j)) = 1$, $i < j$, we will always set $\mathbf{a}^*(j) = \mathbf{a}^*(i) + 1$ since setting $\mathbf{a}^*(i) = \mathbf{a}^*(j) + 1$ results the same computation rate. Then, as long as $\mathbf{h}(i) = \mathbf{h}(j)$, $i < j$, it holds that $\mathbf{a}^*(i) \leq \mathbf{a}^*(j)$.*

Theorem 3. *For a nonnegative ordered channel vector \mathbf{h} , the optimal coefficient vector \mathbf{a}^* is also nonnegative ordered.*

Proof: According to Lemma 2, all the elements in \mathbf{a}^* are nonnegative. Suppose \mathbf{a}^* is not

nonnegative ordered, then there must exist i, j ($1 \leq i < j \leq L$) such that $\mathbf{a}^*(i) > \mathbf{a}^*(j) \geq 0$. According to Lemma 3, $\mathbf{a}^*(i) > 0$ implies $\mathbf{h}(i) > 0$. According to Lemma 4 and Remark 4, $\mathbf{a}^*(i) > \mathbf{a}^*(j)$ implies $\mathbf{h}(i) \neq \mathbf{h}(j)$ and thus $\mathbf{h}(j) > \mathbf{h}(i) > 0$. Then, $\mathbf{h}(i)\mathbf{a}^*(j) + \mathbf{h}(j)\mathbf{a}^*(i) > \mathbf{h}(i)\mathbf{a}^*(i) + \mathbf{h}(j)\mathbf{a}^*(j)$.

Define \mathbf{a}' as: $\mathbf{a}'(i) = \mathbf{a}^*(j)$, $\mathbf{a}'(j) = \mathbf{a}^*(i)$, and $\mathbf{a}'(\ell) = \mathbf{a}^*(\ell)$, $\forall \ell \notin \{i, j\}$. Obviously, $\|\mathbf{a}'\| = \|\mathbf{a}^*\|$, and $\mathbf{h}^T \mathbf{a}' = \sum_{\ell=1}^L \mathbf{h}(\ell) \mathbf{a}'(\ell) = \sum_{\ell \notin \{i, j\}} \mathbf{h}(\ell) \mathbf{a}'(\ell) + \mathbf{h}(i) \mathbf{a}'(i) + \mathbf{h}(j) \mathbf{a}'(j) = \sum_{\ell \notin \{i, j\}} \mathbf{h}(\ell) \mathbf{a}^*(\ell) + \mathbf{h}(i) \mathbf{a}^*(j) + \mathbf{h}(j) \mathbf{a}^*(i) > \sum_{\ell \notin \{i, j\}} \mathbf{h}(\ell) \mathbf{a}^*(\ell) + \mathbf{h}(i) \mathbf{a}^*(i) + \mathbf{h}(j) \mathbf{a}^*(j) = \sum_{\ell=1}^L \mathbf{h}(\ell) \mathbf{a}^*(\ell) = \mathbf{h}^T \mathbf{a}^* \geq 0$. Then according to (2), $\mathcal{R}(\mathbf{h}, \mathbf{a}') > \mathcal{R}(\mathbf{h}, \mathbf{a}^*)$, which implies \mathbf{a}^* is not optimal. Therefore, \mathbf{a}^* must be nonnegative ordered. ■

B. Relaxation to QPs

As stated before, it suffices to obtain the optimal coefficient vector for a nonnegative ordered channel vector. Thus, in the following, we will focus on solving the problem in (4) for a nonnegative ordered channel vector $\bar{\mathbf{h}}$. We first relax this problem to a series of QPs.

Denote the optimal coefficient vector for $\bar{\mathbf{h}}$ as $\bar{\mathbf{a}}^*$. According to Theorem 3, the maximum element in $\bar{\mathbf{a}}^*$ is $\bar{\mathbf{a}}^*(L)$. Suppose $\bar{\mathbf{a}}^*(L)$ is known to be $\bar{a}_L^* \in \mathbb{Z} \setminus \{0\}$, then the problem in (4) can be relaxed as a QP,

$$\begin{aligned} & \underset{\mathbf{a}}{\text{minimize}} && \mathbf{a}^T \mathbf{G} \mathbf{a} \\ & \text{subject to} && \mathbf{a} \in \mathbb{R}^L \\ & && \mathbf{a}(L) = \bar{a}_L^* \end{aligned} \tag{6}$$

where \mathbf{G} is as defined in (5). The problem is convex since \mathbf{G} is positive definite. Denote the solution of this relaxed problem as $\bar{\mathbf{a}}^\dagger \in \mathbb{R}^L$. The intuition behind this relaxation is that appropriate quantization of the real-valued optimal $\bar{\mathbf{a}}^\dagger$ with the constraint $\mathbf{a}(L) = \bar{a}_L^*$ will lead to the integer-valued optimal $\bar{\mathbf{a}}^*$ or at least a close-to-optimal one with a high probability.

However, since $\bar{\mathbf{a}}^*(L)$ is unknown, we alternatively approximate the problem in (6) by solving a series of QPs, i.e., solving the following QP multiple times for $k = 1, 2, \dots, K$.

$$\begin{aligned} & \underset{\mathbf{a}}{\text{minimize}} && \mathbf{a}^T \mathbf{G} \mathbf{a} \\ & \text{subject to} && \mathbf{a} \in \mathbb{R}^L \\ & && \mathbf{a}(L) = k \end{aligned} \tag{7}$$

Denote the solution to the above QP with the constraint $\mathbf{a}(L) = k$ as $\bar{\mathbf{a}}_k^\dagger$. For simplicity, we use $\{s_k\}$ to denote the set with elements being $s_k, k = 1, 2, \dots, K$ in the following. As long as $K \geq \bar{\mathbf{a}}^*(L)$, the solution $\bar{\mathbf{a}}^\dagger$ to the QP in (6) will be included in the set of solutions $\{\bar{\mathbf{a}}_k^\dagger\}$ to the QPs in (7). Fortunately, to obtain the solution set $\{\bar{\mathbf{a}}_k^\dagger\}$, it is sufficient to solve merely one QP in (7) with $k = 1$, according to the following theorem.

Theorem 4. *Denote the solution to the QP in (7) with the constraint $\mathbf{a}(L) = k$ as $\bar{\mathbf{a}}_k^\dagger$, then $\bar{\mathbf{a}}_k^\dagger = k\bar{\mathbf{a}}_1^\dagger$.*

Proof:

$$\begin{aligned}
 \bar{\mathbf{a}}_k^\dagger &= \arg \min_{\mathbf{a} \in \mathbb{R}^L, \mathbf{a}(L)=k} \mathbf{a}^T \mathbf{G} \mathbf{a} \\
 &= k \left(\arg \min_{\mathbf{t} \in \mathbb{R}^L, \mathbf{t}(L)=1} (k\mathbf{t})^T \mathbf{G} (k\mathbf{t}) \right) \quad (\text{Let } \mathbf{a} = k\mathbf{t}.) \\
 &= k \left(\arg \min_{\mathbf{t} \in \mathbb{R}^L, \mathbf{t}(L)=1} k^2 \mathbf{t}^T \mathbf{G} \mathbf{t} \right) \\
 &= k \left(\arg \min_{\mathbf{t} \in \mathbb{R}^L, \mathbf{t}(L)=1} \mathbf{t}^T \mathbf{G} \mathbf{t} \right) \\
 &= k\bar{\mathbf{a}}_1^\dagger.
 \end{aligned}$$

■

The closed-form expression of $\bar{\mathbf{a}}_1^\dagger$ can be readily obtained by solving a linear system as stated in the following theorem, which is the key for the low complexity of our method.

Theorem 5. *Let $\bar{\mathbf{a}}_1^\dagger$ be the optimal solution to the QP in (7) with the constraint $\mathbf{a}(L) = 1$, then*

$$\bar{\mathbf{a}}_1^\dagger = \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix},$$

where

$$\mathbf{r} = -\left(\mathbf{G}(1:L-1, 1:L-1)\right)^{-1} \mathbf{G}(1:L-1, L). \quad (8)$$

Proof: The QP in (7) has only an equality constraint, and thus is linear and particularly simple [15]. We now derive the closed-form solution with the Lagrange multiplier method. Let the Lagrange multiplier associated with the constraint $\mathbf{a}(L) = 1$ be $\lambda \geq 0$, then the Lagrangian

is

$$\mathcal{L}(\mathbf{a}, \lambda) = \mathbf{a}^T \mathbf{G} \mathbf{a} + \lambda (\mathbf{a}(L) - 1).$$

The optimal solution can be obtained by letting the derivative of the Lagrangian be zero, i.e.,

$$\frac{\partial}{\partial \mathbf{a}} \mathcal{L}(\mathbf{a}, \lambda) = (\mathbf{G} + \mathbf{G}^T) \mathbf{a} + \begin{bmatrix} 0 \\ \lambda \end{bmatrix} = 2\mathbf{G} \mathbf{a} + \begin{bmatrix} 0 \\ \lambda \end{bmatrix} = \mathbf{0}.$$

Let $\mathbf{r} = \mathbf{a}(1:L-1)$, $\lambda = 2\mu$, and write \mathbf{G} and \mathbf{a} as block matrices, then

$$\left[\begin{array}{c|c} \mathbf{G}(1:L-1, 1:L-1) & \mathbf{G}(1:L-1, L) \\ \hline \mathbf{G}(L, 1:L-1) & \mathbf{G}(L, L) \end{array} \right] \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mu \end{bmatrix} = \mathbf{0}.$$

In the above equation, observe that

$$\left[\begin{array}{c|c} \mathbf{G}(1:L-1, 1:L-1) & \mathbf{G}(1:L-1, L) \end{array} \right] \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = \mathbf{0},$$

then

$$\mathbf{r} = -\left(\mathbf{G}(1:L-1, 1:L-1)\right)^{-1} \mathbf{G}(1:L-1, L),$$

and the results follow immediately. ■

Calculating $\bar{\mathbf{a}}_1^\dagger$ in Theorem 5 has a complexity order of $O(L^3)$ due to the matrix inversion in the expression of \mathbf{r} . We note that this complexity order can be reduced to $O(L)$ by the following lemma.

Lemma 5. *Equation (8) can be expressed in a simpler form as*

$$\mathbf{r} = \frac{\mathbf{u}(L)}{1 - \|\mathbf{u}(1:L-1)\|^2} \mathbf{u}(1:L-1),$$

where the “normalized” channel vector \mathbf{u} is defined as

$$\mathbf{u} = \sqrt{\frac{P}{1 + P \|\bar{\mathbf{h}}\|^2}} \bar{\mathbf{h}}. \tag{9}$$

Proof: Express \mathbf{G} in (5) and \mathbf{r} in (8) in terms of \mathbf{u} ,

$$\mathbf{G} = \mathbf{I} - \mathbf{u}\mathbf{u}^T,$$

$$\mathbf{r} = (\mathbf{I}^{L-1} - \mathbf{u}(1:L-1)\mathbf{u}^T(1:L-1))^{-1}\mathbf{u}(L)\mathbf{u}(1:L-1),$$

where \mathbf{I}^{L-1} denotes the identity matrix with dimension $L-1$. Then it is easy to verify that Lemma 5 holds. \blacksquare

With Theorems 4 and 5, the K solutions $\{\bar{\mathbf{a}}_k^\dagger\}$ to the K QPs in (7) can be easily obtained.

The next step is to quantize the real-valued approximations $\{\bar{\mathbf{a}}_k^\dagger\}$ to integer vectors by applying the floor or the ceiling functions to each of the elements. One issue that still remains is how to determine the value of K . Intuitively, the larger K , the better. Actually, it is sufficient to set K as

$$K = \arg \max_{\|\lfloor \bar{\mathbf{a}}_k^\dagger \rfloor\|^2 < 1+P\|\bar{\mathbf{h}}\|^2} k \quad (10)$$

according to the following lemma from [2].

Lemma 6. *For a given channel vector \mathbf{h} , the computation rate $\mathcal{R}(\mathbf{h}, \mathbf{a})$ is zero if the coefficient vector \mathbf{a} satisfies*

$$\|\mathbf{a}\|^2 \geq 1 + P\|\mathbf{h}\|^2. \quad (11)$$

Remark 5. *For high SNR (i.e., large P) and large dimensions of $\bar{\mathbf{h}}$, K in (10) can be quite huge. However, as we will show in the next section, for i.i.d. Gaussian channel entries with high SNR, K can be set to a rather small value without degrading the average computation rate.*

In practice, for i.i.d. Gaussian channel entries, we set an upper bound for K as K_u , which is determined off-line according to the simulation results, such that the simulated average computation rate at 20dB with K being K_u is greater than 99% of that with K being $K_u + 1$. We set K_u based on rates at 20dB since the value of K influences more the rates at larger SNR, and 20dB is the the maximum SNR considered in this paper. Then, we set K as the maximum integer that is no greater than K_u while satisfies (10) at the same time, i.e.,

$$K = \arg \max_{\substack{\|\lfloor \bar{\mathbf{a}}_k^\dagger \rfloor\|^2 < 1+P\|\bar{\mathbf{h}}\|^2 \\ k \leq K_u}} k. \quad (12)$$

For implementation, K can be easily determined by using a bi-section search.

C. Quantization

We propose the *successive quantization* algorithm shown in Algorithm 1 to quantize the K real-valued approximations $\{\bar{\mathbf{a}}_k^\dagger\}$ to integer-valued vectors $\{\bar{\mathbf{a}}_k^\diamond\}$ that serve as candidates of a suboptimal coefficient vector $\bar{\mathbf{a}}^\diamond$. For convenience, define

$$f(\mathbf{w}) \triangleq \mathbf{w}^T \mathbf{G} \mathbf{w}, \quad (13)$$

where $\mathbf{w} \in \mathbb{R}^L$, and \mathbf{G} is defined in (5) with \mathbf{h} being nonnegative ordered. Also, let $\lfloor \mathbf{w} \rfloor_\ell$ and $\lceil \mathbf{w} \rceil_\ell$ be the vectors generated from \mathbf{w} by applying the floor and the ceiling operations on the ℓ -th element only, respectively.

<p>Input : A real-valued vector $\bar{\mathbf{a}}_k^\dagger \in \mathbb{R}^L$ Output: A coefficient vector $\bar{\mathbf{a}}_k^\diamond \in \mathbb{Z}^L$ for $\bar{\mathbf{h}}$</p> <pre> 1 for $\ell \leftarrow 1$ to $L - 1$ do 2 if $f\left(\lfloor \bar{\mathbf{a}}_k^\dagger \rfloor_\ell\right) < f\left(\lceil \bar{\mathbf{a}}_k^\dagger \rceil_\ell\right)$ then 3 $\bar{\mathbf{a}}_k^\dagger \leftarrow \lfloor \bar{\mathbf{a}}_k^\dagger \rfloor_\ell$ 4 else 5 $\bar{\mathbf{a}}_k^\dagger \leftarrow \lceil \bar{\mathbf{a}}_k^\dagger \rceil_\ell$ 6 end 7 end 8 $\bar{\mathbf{a}}_k^\diamond \leftarrow \bar{\mathbf{a}}_k^\dagger$ 9 return $\bar{\mathbf{a}}_k^\diamond$ </pre>
--

Algorithm 1. Successive Quantization

To simplify the inequality condition $f\left(\lfloor \bar{\mathbf{a}}_k^\dagger \rfloor_\ell\right) < f\left(\lceil \bar{\mathbf{a}}_k^\dagger \rceil_\ell\right)$ in line 2 of Algorithm 1, we first introduce the following lemma.

Lemma 7. For the function $f(\mathbf{w}) = \mathbf{w}^T \mathbf{G} \mathbf{w}$ defined in (13) where $\mathbf{G}^T = \mathbf{G}$, the inequality condition $f(\lfloor \mathbf{w} \rfloor_\ell) < f(\lceil \mathbf{w} \rceil_\ell)$ is equivalent to

$$2(\lfloor \mathbf{w} \rfloor_\ell)^T \mathbf{G}(:, \ell) + \mathbf{G}(\ell, \ell) > 0. \quad (14)$$

Proof: $f(\lfloor \mathbf{w} \rfloor_\ell) < f(\lceil \mathbf{w} \rceil_\ell)$ implies $\lfloor \mathbf{w} \rfloor_\ell \neq \lceil \mathbf{w} \rceil_\ell$, i.e., $\mathbf{w}(\ell)$ is not an integer. Let $\mathbf{e}_\ell \in \mathbb{R}^L$ be the vector with only one nonzero element $\mathbf{e}_\ell(\ell) = 1$, then $\lceil \mathbf{w} \rceil_\ell = \lfloor \mathbf{w} \rfloor_\ell + \mathbf{e}_\ell$, and

$$\begin{aligned}
f(\lceil \mathbf{w} \rceil_\ell) &= (\lceil \mathbf{w} \rceil_\ell)^T \mathbf{G} \lceil \mathbf{w} \rceil_\ell \\
&= (\lfloor \mathbf{w} \rfloor_\ell + \mathbf{e}_\ell)^T \mathbf{G} (\lfloor \mathbf{w} \rfloor_\ell + \mathbf{e}_\ell) \\
&= (\lfloor \mathbf{w} \rfloor_\ell)^T \mathbf{G} \lfloor \mathbf{w} \rfloor_\ell + (\lfloor \mathbf{w} \rfloor_\ell)^T \mathbf{G} \mathbf{e}_\ell + \mathbf{e}_\ell^T \mathbf{G} \lfloor \mathbf{w} \rfloor_\ell + \mathbf{e}_\ell^T \mathbf{G} \mathbf{e}_\ell \\
&= f(\lfloor \mathbf{w} \rfloor_\ell) + (\lfloor \mathbf{w} \rfloor_\ell)^T \mathbf{G}(:, \ell) + \mathbf{G}(\ell, :) \lfloor \mathbf{w} \rfloor_\ell + \mathbf{G}(\ell, \ell) \\
&= f(\lfloor \mathbf{w} \rfloor_\ell) + 2(\lfloor \mathbf{w} \rfloor_\ell)^T \mathbf{G}(:, \ell) + \mathbf{G}(\ell, \ell).
\end{aligned}$$

Obviously, $f(\lfloor \mathbf{w} \rfloor_\ell) < f(\lceil \mathbf{w} \rceil_\ell)$ is equivalent to

$$2(\lfloor \mathbf{w} \rfloor_\ell)^T \mathbf{G}(:, \ell) + \mathbf{G}(\ell, \ell) > 0.$$

■

Lemma 8. *With Lemma 7, the inequality condition $f(\lfloor \bar{\mathbf{a}}_k^\dagger \rfloor_\ell) < f(\lceil \bar{\mathbf{a}}_k^\dagger \rceil_\ell)$ in line 2 of Algorithm 1 can be simplified as*

$$2 \lfloor \bar{\mathbf{a}}_k^\dagger(\ell) \rfloor - 2 \left(\left(\lfloor \bar{\mathbf{a}}_k^\dagger \rfloor_\ell \right)^T \mathbf{u} \right) \mathbf{u}(\ell) + 1 - \mathbf{u}(\ell)^2 < 0,$$

where \mathbf{u} is the normalized channel vector as defined in (9).

Proof: The proof is straightforward by writing \mathbf{G} in terms of \mathbf{u} , and thus is omitted here.

■

After the quantization, a suboptimal coefficient vector $\bar{\mathbf{a}}^\diamond$ for $\bar{\mathbf{h}}$ is obtained with

$$\bar{\mathbf{a}}^\diamond = \arg \min_{\mathbf{a} \in \{\bar{\mathbf{a}}_k^\diamond\}} \mathbf{a}^T \mathbf{G} \mathbf{a}. \quad (15)$$

Finally, a suboptimal coefficient vector \mathbf{a}^\diamond for the original channel vector \mathbf{h} is recovered from $\bar{\mathbf{a}}^\diamond$ according to Remark 3.

We summarize our proposed *QP relaxation* method in Algorithm 2. The pseudocode is shown in Algorithm 3, where the function $[\bar{\mathbf{w}}, \mathbf{p}] = \text{sort}(\mathbf{w})$ sorts the elements in \mathbf{w} in ascending order, returns the sorted vector $\bar{\mathbf{w}}$, and stores the original indices of the elements as vector \mathbf{p} , the function $\text{floor}(\mathbf{w})$ applies the floor operation to each element of \mathbf{w} and returns the resulted

integer vector.

Input : A channel vector $\mathbf{h} \in \mathbb{R}^L$, power P ,
an upper bound K_u (determined off-line) for K

Output: A coefficient vector $\mathbf{a}^\diamond \in \mathbb{Z}^L$ for \mathbf{h}

- 1) Preprocess \mathbf{h} to the nonnegative ordered $\bar{\mathbf{h}}$ with Remark 3.
- 2) Calculate $\bar{\mathbf{a}}_1^\dagger$, with Theorem 5 and Lemma 5.
- 3) Determine K with (12).
- 4) Calculate $\{\bar{\mathbf{a}}_k^\dagger\}$, i.e., the real-valued approximations of the optimal coefficient vector for $\bar{\mathbf{h}}$, using Theorem 4.
- 5) Quantize $\{\bar{\mathbf{a}}_k^\dagger\}$ to integer-valued vectors $\{\bar{\mathbf{a}}_k^\diamond\}$ with Algorithm 1.
- 6) Select a vector from $\{\bar{\mathbf{a}}_k^\diamond\}$ to be a suboptimal coefficient vector $\bar{\mathbf{a}}^\diamond$ for $\bar{\mathbf{h}}$ using (15).
- 7) Recover a suboptimal coefficient vector \mathbf{a}^\diamond for \mathbf{h} from $\bar{\mathbf{a}}^\diamond$ according to Remark 3.

Algorithm 2. The Proposed QP Relaxation Method – Outline

D. Complexity Analysis

Here we analyze the complexity of our algorithm, in terms of the number of flops required. Referring to the outline in Algorithm 2, the processing of \mathbf{h} in step 1 involves recording the signs of the elements and sorting the elements, and takes $O(L \log(L))$ flops. Calculating $\bar{\mathbf{a}}_1^\dagger$ in step 2 has a complexity of $O(L)$. For the bi-section search applied to determine K in step 3, the maximum number of loops required to execute is $O(\log(K_u))$, the number of flops in each loop is $O(L)$, and thus the maximum cost is $O(\log(K_u)L)$. Step 4 takes $O(KL)$ flops. By introducing appropriate temporary variables b and d as shown in Algorithm 3, the successive quantization of a real-valued approximation $\bar{\mathbf{a}}_k^\dagger$ can be implemented in an efficient way in $O(L)$ flops. Thus, the complexity of quantizing all the K real-valued approximations is $O(KL)$. Selecting a coefficient vector from the quantized vector set in step 6 has a cost of $O(KL)$. Step 7 takes $O(L)$ flops. In summary, the complexity of the method is $O(L(\log(L) + \log(K_u) + K))$.

However, the above analyzed complexity expression involves the experiment-based K_u , and its exact order with respect to the dimension L is intractable. As an alternative, we use an upper bound to approximate the cost. According to (10), it is easy to see that K and K_u are at most of order $O(\sqrt{P \|\mathbf{h}\|^2})$. Then, the complexity of our method is $O(L(\log(L) + \sqrt{P \|\mathbf{h}\|^2}))$. We reserve the power P in the expression since we may also care about how the complexity varies when the SNR gets large.

Input : A channel vector $\mathbf{h} \in \mathbb{R}^L$, power P ,
an upper bound K_u (determined off-line) for K
Output: A coefficient vector $\mathbf{a}^\diamond \in \mathbb{Z}^L$ for \mathbf{h}

```

1 // Preprocessing
2  $\mathbf{t} \leftarrow \text{sign}(\mathbf{h})$  // Get signs of entries in  $\mathbf{h}$ .
3 // Sort  $\text{abs}(\mathbf{h})$  in ascending order as  $\bar{\mathbf{h}}$ .
4 //  $\mathbf{p}$  stores original indices.
5  $(\bar{\mathbf{h}}, \mathbf{p}) \leftarrow \text{sort}(\text{abs}(\mathbf{h}))$ 
6  $b \leftarrow 1 + P \|\mathbf{h}\|^2$  // A constant for efficiency
7 // Calculate  $\bar{\mathbf{a}}_1^\dagger$ .
8  $\mathbf{u} \leftarrow (P/b)^{1/2} \bar{\mathbf{h}}$  // Normalized channel vector
9  $\mathbf{r} \leftarrow \frac{\mathbf{u}(L)}{1 - \|\mathbf{u}(1:L-1)\|^2} \mathbf{u}(1:L-1)$ 
10  $\bar{\mathbf{a}}_1^\dagger(1:L-1) \leftarrow \mathbf{r}$ 
11  $\bar{\mathbf{a}}_1^\dagger(L) \leftarrow 1$ 
12 // Determine  $K$ .
13 if  $\|\text{floor}(K_u \bar{\mathbf{a}}_1^\dagger)\|^2 < b$  then
14    $K \leftarrow K_u$ 
15 else // Bi-section search
16    $K_l \leftarrow 1$ 
17   while  $K_u \neq K_l + 1$  do
18      $K \leftarrow \text{floor}((K_u + K_l)/2)$ 
19     if  $\|\text{floor}(K \bar{\mathbf{a}}_1^\dagger)\|^2 < b$  then
20        $K_l \leftarrow K$ 
21     else
22        $K_u \leftarrow K$ 
23     end
24   end
25    $K \leftarrow K_l$ 
26 end
27 // Quantization
28  $\bar{\mathbf{a}}^\diamond(1:L-1) \leftarrow \mathbf{0}$  // Initialize  $\bar{\mathbf{a}}^\diamond$  and  $f_{\min}$ .
29  $\bar{\mathbf{a}}^\diamond(L) \leftarrow 1$ 
30  $f_{\min} \leftarrow \|\bar{\mathbf{a}}^\diamond\|^2 - ((\bar{\mathbf{a}}^\diamond)^T \mathbf{u})^2$  //  $f \triangleq \mathbf{a}^T \mathbf{G} \mathbf{a}$ 
31 for  $k \leftarrow 1$  to  $K$  do
32    $\bar{\mathbf{a}}^\dagger \leftarrow k \bar{\mathbf{a}}_1^\dagger$  // Calculate  $\bar{\mathbf{a}}_k^\dagger$ .
33    $d \leftarrow (\bar{\mathbf{a}}^\dagger)^T \mathbf{u}$  // Temporary variable
34   for  $\ell \leftarrow 1$  to  $L-1$  do
35      $v \leftarrow \bar{\mathbf{a}}^\dagger(\ell)$ 
36      $\bar{\mathbf{a}}^\dagger(\ell) \leftarrow \text{floor}(\bar{\mathbf{a}}^\dagger(\ell))$ 
37      $d \leftarrow d + (\bar{\mathbf{a}}^\dagger(\ell) - v) \mathbf{u}(\ell)$ 
38     if  $2\bar{\mathbf{a}}^\dagger(\ell) - 2d\mathbf{u}(\ell) + 1 - \mathbf{u}(\ell)^2 < 0$  then
39        $\bar{\mathbf{a}}^\dagger(\ell) \leftarrow \bar{\mathbf{a}}^\dagger(\ell) + 1$ 
40        $d \leftarrow d + \mathbf{u}(\ell)$ 
41     end
42   end
43   // Update the record.
44    $f \leftarrow \|\bar{\mathbf{a}}^\dagger\|^2 - d^2$ 
45   if  $f < f_{\min}$  then
46      $\bar{\mathbf{a}}^\diamond \leftarrow \bar{\mathbf{a}}^\dagger$ 
47      $f_{\min} \leftarrow f$ 
48   end
49 end
50 // Computation rate is  $1/2 \log(1/f_{\min})$ .
51 // Recover the coefficient vector.
52 for  $\ell \leftarrow 1$  to  $L$  do
53    $\mathbf{a}^\diamond(p(\ell)) \leftarrow \mathbf{t}(p(\ell)) \bar{\mathbf{a}}^\diamond(\ell)$ 
54 end
55 return  $\mathbf{a}^\diamond$ 

```

Algorithm 3. The Proposed QP Relaxation Method – Pseudocode

In the complexity expression above, since the square root function is strictly concave, it follows from Jensen's inequality that $\mathbb{E}(\sqrt{\|\mathbf{h}\|^2}) \leq \sqrt{\mathbb{E}(\|\mathbf{h}\|^2)}$. Specifically, for i.i.d. standard Gaussian channel entries, the expectation of $\|\mathbf{h}\|^2$ is L , and thus the corresponding average complexity of the proposed method becomes $O(L \log(L) + P^{0.5} L^{1.5})$. It is easy to see that the complexity is of order 1.5 with respect to the dimension L .

E. Extension to the Complex-Valued Channel Model

We now consider the complex-valued channel model of the AWGN networks, and demonstrate how to apply the proposed QP relaxation method for complex-valued channels. The complex-valued channel model is defined as below.

Definition 6. (Complex-Valued Channel Model) *In an AWGN network, each relay (indexed by $m = 1, 2, \dots, M$) observes a noisy linear combination of the transmitted signals through the channel,*

$$\mathbf{y}_m = \sum_{\ell=1}^L \mathbf{h}_m(\ell) \mathbf{x}_\ell + \mathbf{z}_m, \quad (16)$$

where $\mathbf{x}_\ell \in \mathbb{C}^n$ with the power constraint $\frac{1}{n} \|\mathbf{x}_\ell\|^2 \leq P$ is the transmitted codeword from source ℓ ($\ell = 1, 2, \dots, L$), $\mathbf{h}_m \in \mathbb{C}^L$ is the channel vector to relay m , $\mathbf{z}_m \in \mathbb{C}^n$ is the noise vector with entries being i.i.d. Gaussian, i.e., $\mathbf{z}_m \sim \mathcal{CN}(\mathbf{0}, \mathbf{I})$, and \mathbf{y}_m is the signal received at relay m .

Similar to what we have done for the real-valued channel model, we will focus on one relay, and ignore the subscript “ m ” for notational convenience.

Writing the summation in (16) in the vector product form, (16) becomes

$$\mathbf{y} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L] \mathbf{h} + \mathbf{z}. \quad (17)$$

It is well-known that a complex-valued channel model can be written in its real-valued equivalent form. Let $\Re(\mathbf{w})$ denote the vector composed of the real part of \mathbf{w} , and $\Im(\mathbf{w})$ denote the vector composed of the imaginary part of \mathbf{w} . The complex-valued equation (17) has the following real-equivalent form

$$[\Re(\mathbf{y}), \Im(\mathbf{y})] = [\Re(\mathbf{x}_1), \Re(\mathbf{x}_2), \dots, \Re(\mathbf{x}_L), \Im(\mathbf{x}_1), \Im(\mathbf{x}_2), \dots, \Im(\mathbf{x}_L)] \times \begin{bmatrix} \Re(\mathbf{h}), \Im(\mathbf{h}) \\ -\Im(\mathbf{h}), \Re(\mathbf{h}) \end{bmatrix} + [\Re(\mathbf{z}), \Im(\mathbf{z})]. \quad (18)$$

It is obvious that

$$\begin{aligned}\Re(\mathbf{y}) &= [\Re(\mathbf{x}_1), \Re(\mathbf{x}_2), \dots, \Re(\mathbf{x}_L), \Im(\mathbf{x}_1), \Im(\mathbf{x}_2), \dots, \Im(\mathbf{x}_L)] \times \begin{bmatrix} \Re(\mathbf{h}) \\ -\Im(\mathbf{h}) \end{bmatrix} + \Re(\mathbf{z}), \\ \Im(\mathbf{y}) &= [\Re(\mathbf{x}_1), \Re(\mathbf{x}_2), \dots, \Re(\mathbf{x}_L), \Im(\mathbf{x}_1), \Im(\mathbf{x}_2), \dots, \Im(\mathbf{x}_L)] \times \begin{bmatrix} \Im(\mathbf{h}) \\ \Re(\mathbf{h}) \end{bmatrix} + \Im(\mathbf{z}).\end{aligned}\tag{19}$$

Then we can view $\begin{bmatrix} \Re(\mathbf{h}) \\ -\Im(\mathbf{h}) \end{bmatrix}$ and $\begin{bmatrix} \Im(\mathbf{h}) \\ \Re(\mathbf{h}) \end{bmatrix}$ as two $2L$ -dimensional real-valued channels, and view $\Re(\mathbf{x}_\ell)$ and $\Im(\mathbf{x}_\ell)$ as two independent n -dimensional real-valued transmitted codewords. Assume equal power allocation on the real part and the imaginary part of each transmitted codeword, i.e., $\|\Re(\mathbf{x}_\ell)\|^2 = \|\Im(\mathbf{x}_\ell)\|^2$, then the power constraint of each real-valued transmitted codeword is $\frac{1}{n} \|\Re(\mathbf{x}_\ell)\|^2 \leq \frac{1}{2}P$ and $\frac{1}{n} \|\Im(\mathbf{x}_\ell)\|^2 \leq \frac{1}{2}P$.

Based on the above interpretation, we can apply the proposed QP relaxation method to each of the two $2L$ -dimensional channels to find the corresponding coefficient vectors. It should be noted that we only need to find the coefficient vector for one of the $2L$ -dimensional channels, which saves half of the computation cost. Let \mathbf{a} be a Gaussian integer, and assume the found coefficient vector for $\begin{bmatrix} \Re(\mathbf{h}) \\ -\Im(\mathbf{h}) \end{bmatrix}$ is $\begin{bmatrix} \Re(\mathbf{a}) \\ -\Im(\mathbf{a}) \end{bmatrix}$. Then, according to Theorem 2, the coefficient vector for $\begin{bmatrix} \Im(\mathbf{h}) \\ \Re(\mathbf{h}) \end{bmatrix}$ is $\begin{bmatrix} \Im(\mathbf{a}) \\ \Re(\mathbf{a}) \end{bmatrix}$. In this sense, for each complex-valued channel vector \mathbf{h} , we can find a Gaussian integer as the best coefficient vector \mathbf{a} using the QP relaxation method.

IV. NUMERICAL RESULTS

In this section, we present some numerical results to demonstrate the effectiveness and efficiency of our QP relaxation approach. As explained before, finding the coefficient vector for a complex-valued channel can be transformed to finding the coefficient vector for a real-valued channel. Thus, we focus on the real-valued channels here. We consider the case where the entries of the channel vector \mathbf{h} are i.i.d. standard Gaussian, i.e., $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In our simulations, the dimension L ranges from 2 to 16, and the power P ranges from 0dB to 20dB. For a given dimension and a given power, we randomly generate 10000 instances of the channel vector, and apply the QP relaxation method to find the coefficient vectors, and calculate the corresponding average computation rate.

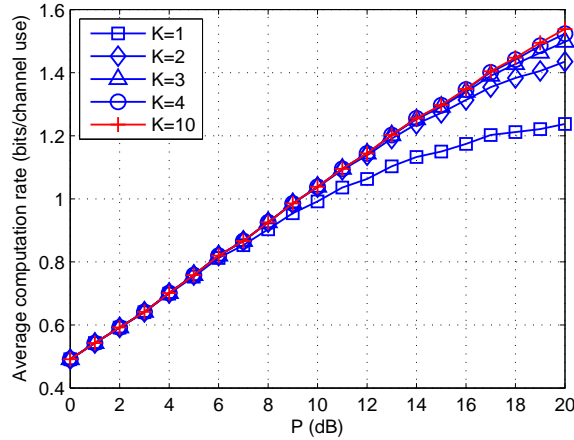


Fig. 1: Average computation rate for $L = 4$ using our QPR method with different K .

We first show that as stated in Remark 5, for high dimension and large power, the number of real-valued approximations K can be set to a rather small value without degrading the rate apparently. As shown in Figure 1, for dimension $L = 4$ and power P from 0dB to 20dB, the average computation rate quickly converges as K increases from 1 to 4. Further increasing K up to 10 incurs additional computational cost with little improvement in the average computation rate.

With the above observation, it is reasonable to introduce the upper bound K_u for K , and adopt the criterion in (12) to determine K . K_u can be calculated off-line by simulations prior to applying the method, which incurs no additional processing complexity in real-time. The values of K_u according to the simulation results are listed in Table I.

TABLE I: K_u in (12) for the proposed QPR method

L	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
K_u	2	3	4	5	5	5	6	6	6	6	7	6	6	6	4

We then show the effectiveness of our method by comparing the average computation rate with those of other existing methods. The methods covered include the following.

- Our QP relaxation (QPR) method that gives the suboptimal solution.
- The branch-and-bound (BnB) method proposed by Richter *et al.* in [4] that provides the optimal solution.

- The method developed by Sahraei and Gastpar in [5] that finds the optimal solution with an average-case complexity of $O(P^{0.5}L^{2.5})$ for i.i.d. Gaussian channel entries. We refer to this method as the “SG” method for short.
- The LLL method proposed by Sakzad *et al.* in [12], which is based on the LLL lattice reduction (LR) algorithm. The parameter δ in the LLL LR algorithm is set as 0.75 since further increasing δ towards 1 achieves little gain in the computation rate but requires more computation labor. Although the LLL LR algorithm has known average complexity for some cases [16], [17], its average complexity for our case is unknown, and the worst-case

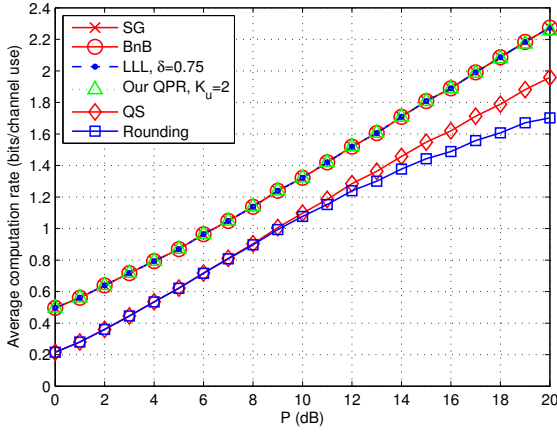
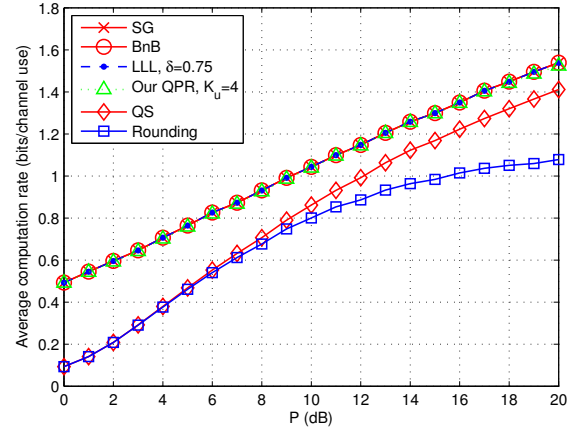
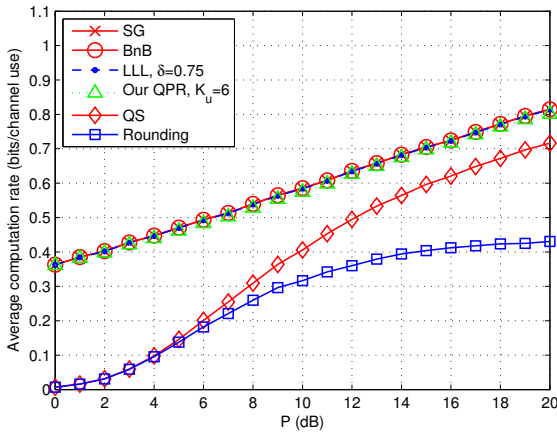
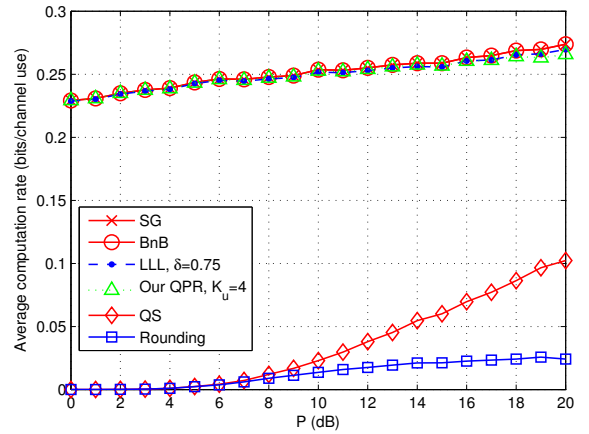
(a) $L = 2$ (b) $L = 4$ (c) $L = 8$ (d) $L = 16$

Fig. 2: Average computation rate using different methods.

complexity could be unbounded [18].

- The quantized search (QS) method developed by Sakzad *et al.* in [12]. The search consists of two phases: 1) an integer α_0 between 1 and $\lfloor P^{1/2} \rfloor$ that provides the maximum rate is selected as the initial value of the amplifying factor α ; 2) the amplifying factor is then refined by searching in $[\alpha_0 - 1, \alpha_0 + 1]$ with a step size 0.1. After the amplifying factor α is determined, the coefficient vector \mathbf{a} is set as $\lfloor \alpha \mathbf{h} \rfloor$. An improved version of the QS method is the quantized exhaustive search (QES) method proposed in [19], which was developed for complex-valued channels.
- The rounding method that simply sets the coefficient vector by rounding the channel vector to an integer-valued vector.

As shown in Figure 2, the optimal methods, i.e., the BnB method and the SG method, always provide the highest average computation rates for all dimensions and over the whole SNR regime, as expected. The LLL method provides close-to-optimal average computation rates. Our proposed QPR method also offers close-to-optimal average computation rates for almost all the dimensions and SNR values considered, except that its performance degrades a little bit for high dimensions at high SNR as shown in Figure 2d. The performance of our QPR method improves slightly compared with the version we presented in [13], [14]. The reason is that here we initialize the output coefficient vector as $[0, \dots, 0, 1]^T$, which definitely results non-zero computation rate, while in the previous version the output coefficient vector could result zero computation rate.

Finally, we demonstrate the efficiency of the proposed QPR method by comparing the running time of finding the coefficient vectors for 10000 channel vector samples. The methods considered include those that provide optimal rates and close-to-optimal rates, i.e., the SG method, the BnB method, the LLL method, and our QPR method. The running time varies for different SNR values, and thus we compare the running time with P being 0dB, 10dB and 20dB. As shown in Figure 3, the proposed QPR method is much more efficient than all the other methods, especially for high dimensions. Specifically, the running time of the optimal methods can be one scale larger than that of the QPR method.

In summary, for i.i.d. Gaussian channel entries, our proposed QPR method offers close-to-optimal average computation rates with a much lower complexity than that of the existing optimal and close-to-optimal methods.

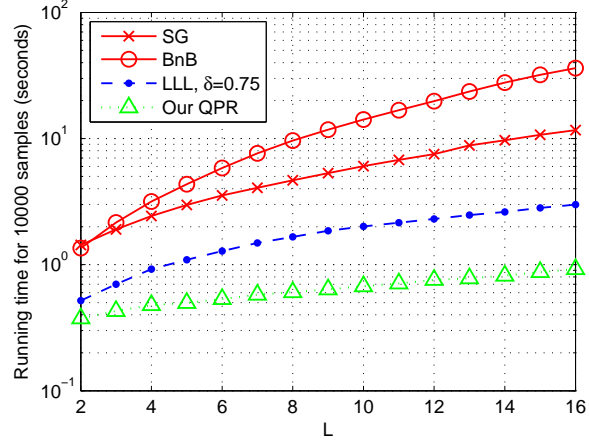
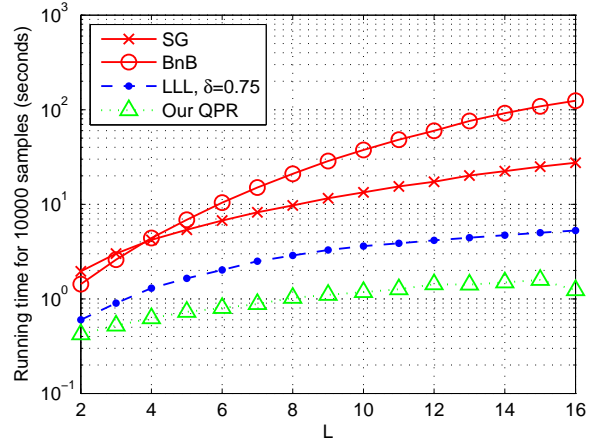
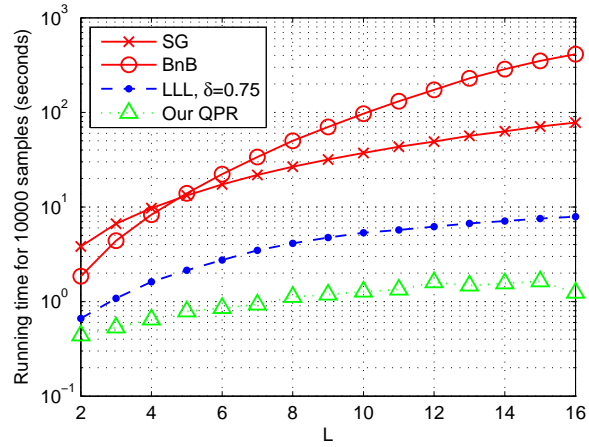
(a) $P = 0$ dB(b) $P = 10$ dB(c) $P = 20$ dB

Fig. 3: Running time for 10000 samples using different methods.

V. CONCLUSIONS

In this paper, we considered the compute-and-forward network coding design problem of finding the optimal coefficient vector that maximizes the computation rate at a relay, and developed the quadratic programming (QP) relaxation method that finds a high quality suboptimal solution. We first revealed some useful properties of the problem, and relaxed the problem to a series of QPs. We then derived the closed-form solutions of the QPs, which is the key to the efficiency of our method, and proposed a successive quantization algorithm to quantize the real-valued solutions to integer vectors that serve as candidates of the coefficient vector. Finally, the candidate that maximizes the computation rate is selected as the best coefficient vector. For L -dimensional channel vectors with i.i.d. Gaussian entries, the average-case complexity of the proposed QP relaxation method is of order 1.5 with respect to the dimension L . Numerical results demonstrated that our QP relaxation method offers close-to-optimal computation rates, and is much more computationally efficient than the existing methods that provide the optimal computation rates as well as the LLL method that also provides close-to-optimal computation rates.

REFERENCES

- [1] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference with structured codes," in *2008 IEEE International Symposium on Information Theory (ISIT)*, 2008, pp. 772–776.
- [2] —, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6463–6486, Oct 2011.
- [3] L. Wei and W. Chen, "Compute-and-forward network coding design over multi-source multi-relay channels," *IEEE Transactions on Wireless Communications*, vol. 11, no. 9, pp. 3348–3357, 2012.
- [4] J. Richter, C. Scheunert, and E. Jorswieck, "An efficient branch-and-bound algorithm for compute-and-forward," in *2012 IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2012, pp. 77–82.
- [5] S. Sahraei and M. Gastpar, "Compute-and-forward: Finding the best equation," in *to appear in 52nd Annual Allerton Conference on Communication, Control, and Computing, Champaign, Illinois, USA*, 2014.
- [6] W. Zhang, S. Qiao, and Y. Wei, "HKZ and Minkowski reduction algorithms for lattice-reduction-aided MIMO detection," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 5963–5976, 2012.
- [7] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring polynomials with rational coefficients." *Mathematische Annalen*, vol. 261, pp. 515–534, 1982. [Online]. Available: <http://eudml.org/doc/182903>
- [8] H. Vetter, V. Ponnampalam, M. Sandell, and P. Hoeher, "Fixed complexity LLL algorithm," *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1634–1637, April 2009.

- [9] C. Ling, W. H. Mow, and N. Howgrave-Graham, "Reduced and fixed-complexity variants of the LLL algorithm for communications," *IEEE Transactions on Communications*, vol. 61, no. 3, pp. 1040–1050, 2013.
- [10] X.-W. Chang, J. Wen, and X. Xie, "Effects of the LLL reduction on the success probability of the Babai point and on the complexity of sphere decoding," *IEEE Transactions on Information Theory*, vol. 59, no. 8, pp. 4915–4926, Aug 2013.
- [11] Y. H. Gan, C. Ling, and W. H. Mow, "Complex lattice reduction algorithm for low-complexity full-diversity mimo detection," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2701–2710, 2009.
- [12] A. Sakzad, E. Viterbo, Y. Hong, and J. Boutros, "On the ergodic rate for compute-and-forward," in *2012 International Symposium on Network Coding (NetCod)*, 2012, pp. 131–136.
- [13] J. Wen, B. Zhou, W. H. Mow, and X.-W. Chang, "Compute-and-forward protocol design based on improved sphere decoding," in *(to appear) Proceedings of IEEE International Conference on Communications*, 2015.
- [14] B. Zhou and W. H. Mow, "A quadratic programming relaxation approach to compute-and-forward network coding design," in *2014 IEEE International Symposium on Information Theory (ISIT)*, June 2014, pp. 2296–2300.
- [15] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 3rd ed. US: Springer, 2008.
- [16] H. Daude and B. Vallee, "An upper bound on the average number of iterations of the LLL algorithm," *Theoretical Computer Science*, vol. 123, no. 1, pp. 95 – 115, 1994. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/030439759490071X>
- [17] C. Ling and N. Howgrave-Graham, "Effective LLL reduction for lattice decoding," in *2007 IEEE International Symposium on Information Theory (ISIT)*, June 2007, pp. 196–200.
- [18] J. Jalden, D. Seethaler, and G. Matz, "Worst- and average-case complexity of LLL lattice reduction in mimo wireless systems," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 2685–2688.
- [19] A. Sakzad, E. Viterbo, J. J. Boutros, and Y. Hong, "Phase precoding for the compute-and-forward protocol," *CoRR*, vol. abs/1404.4157, 2014. [Online]. Available: <http://arxiv.org/abs/1404.4157>